IN THE CLAIMS:

1. (Previously Presented) A method for compressing an input string, comprising the steps of:

generating a lexicographic normal form from said input string, using only a single pass over said input string, wherein said input string has symbols belonging to a partially commutative alphabet; and

applying a compression scheme to said lexicographic normal form.

2. (Original) The method of claim 1, wherein said compression scheme is a grammar-based lossless data compression scheme.

3. (Original) The method of claim 1, wherein said input string is one or more program instructions.

4. (Original) The method of claim 1, wherein said input string is one or more events in a communications network.

5. (Original) The method of claim 1, wherein said generating step further comprises the step of evaluating a set of equivalent words with respect to a noncommutation graph.

6. (Original) The method of claim 1, wherein said generating step further comprises the steps of:

employing a stack corresponding to each vertex $v \in V$, where w is a word over an alphabet V;

processing symbols of w from right to left;

upon seeing a letter u, pushing a u on its stack and a marker pushed on the stacks corresponding to symbols which are adjacent to u in a noncommutation graph G; and

once the entire word has been processed, using said stacks to determine said lexicographic normal form for an interchange class containing the word.

7. (Previously Presented) A method for compressing an input string, comprising the steps of:

generating a Foata normal form from said input string, wherein said input string has symbols belonging to a partially commutative alphabet; and

applying a compression scheme to said Foata normal form.

8. (Original) The method of claim 7, wherein said compression scheme is a grammar-based lossless data compression scheme.

9. (Original) The method of claim 7, wherein said input string is one or more program instructions.

10. (Original) The method of claim 7, wherein said input string is one or more events in a communications network.

11. (Original) The method of claim 7, wherein said generating step further comprises the step of evaluating a set of equivalent words with respect to a noncommutation graph.

12. (Original) The method of claim 7, wherein said generating step further comprises the steps of:

employing a stack corresponding to each vertex $v \in V$, where w is a word over an alphabet V;

processing symbols of w from right to left;

upon seeing a letter u, pushing a u on its stack and a marker on the stacks corresponding to symbols which are adjacent to u in a noncommutation graph G; and

once the entire word has been processed, using said stacks to determine said Foata normal form for an interchange class containing the word.

13. (Previously Presented) A compression system, comprising:

a memory; and

a processor operatively coupled to said memory, said processor configured to:

generate a normal form from said input string, using only a single pass over said input string, wherein said input string has symbols belonging to a partially commutative alphabet; and

applying a compression scheme to said normal form.

14. (Original) The compression system of claim 13, wherein said compression scheme is a grammar-based lossless data compression scheme.

15. (Original) The compression system of claim 13, wherein said input string is one or more program instructions.

16. (Original) The compression system of claim 13, wherein said input string is one or more events in a communications network.

17. (Original) The compression system of claim 13, wherein said normal form is a lexicographic normal form.

18. (Original) The compression system of claim 13, wherein said normal form is a Foata normal form.

19. (Original) The compression system of claim 13, wherein said wherein said processor is further configured to evaluate a set of equivalent words with respect to a noncommutation graph.

20. (Original) The compression system of claim 13, wherein said wherein said processor is further configured to:

employ a stack corresponding to each vertex $v \in V$, where w is a word over an alphabet V;

process symbols of w from right to left;

upon seeing a letter u, pushing a u on its stack and a marker on the stacks corresponding to symbols which are adjacent to u in the noncommutation graph G; and

once the entire word has been processed, using said stacks to determine said normal form for an interchange class containing the word.